COMP123-03 Final Project: Reversi

Julia Zhong & Jacqueline Ong | Fall 2018

User's Manual - Reversi

Our program is a recreation of the popular two-player board game "Reversi", also known as Othello. The objective of Reversi is for each player to finish with more discs of his or her own color on the board than the opponent. This is achieved by strategically placing discs on the board to trap one of the opponent's discs, causing it to flip to the current player's color.

Traditionally, Reversi is played on an 8*8 board with 64 squares (*Jokosare, S.L., Reversi - Game Rules, (n.d.)*), however, we have opted to scale our project down to a 6*6, 36-square board. We believe that the slightly smaller grid size makes our program and code more manageable, and it simplifies the game for first-time players while still allowing for fast-paced and challenging gameplay. Another unique feature of our program is the ability to let players choose their own disc and grid colors, whereas traditional Reversi utilizes black and white discs.

Our entire game can be launched directly by running the file *reversi.py* from where it has been downloaded to in your computer's directory (running through Pycharm is unnecessary for our program!). You will not need to install additional packages or files, as long as Python and the `tkinter` module are installed .

When you run our program, you will see the main window, which contains the grid on which you will play and various buttons that you can interact with. Located above the grid, the 'Choose Colors!' button allows you to set disc colors for both players and the color of the grid. You don't have to change the colors but if you want to, make sure you make no typos, fill all the fields and to click on the 'Start' button after typing the colors. See what colors you can have here (*Python Color Constants Module, 2015)*, listed alphabetically (we did not add this link to the in-game Instructions window because you are unable to copy directly from that window). Three more buttons - 'Instructions', 'Reset' and 'Quit' - are located to the left of the grid. Clicking the 'Instructions' button will bring up a new window with the game rules and a few basic pointers on how to use our program. The 'Reset' button will wipe the current

game and set the grid and discs back to their starting positions, and can be clicked at any time. The 'Quit' button will stop the current game and exit out of the entire program immediately.

To begin gameplay, you can click on any of the empty squares on the grid. Player 1 (black by default) always goes first. Players take turns placing their discs on the grid, and discs will automatically flip and change colors when a valid move has been made by the current player; that is, if they have managed to trap one of the opponent's discs between their own discs in a vertical, horizontal or diagonal direction. Be careful, though! You may accidentally trap your own disc while trying to flip your opponent's. Gameplay ends when the grid is filled. When the game is over, a new window will popup with both players' scores and stating the winning player. A 'Replay' button will also be available, which restarts the game with a clean grid when clicked. Click on the 'Quit' button if you want to end the game.

AI mode is unavailable on the current version of our game, but you don't necessarily need two players to enjoy the game as it is! Since it is highly possible to trap your own disc by accident, you may find great entertainment in playing the game against yourself!

Program Contents and Summary

❖ Program Structure

Our entire program has been written in a single .py file using the programming environment Pycharm Community 2018.2 and the `tkinter` module for GUIs.

The structure of our code can be split into three sections, adapted from the Fox tkinter reading document (*Fox, The tkinter module for Graphical User Interfaces (GUIs), 2018)* that was used in this course. Accompanying the name of each method defined in the second section is a very short description of its purpose.

```
# ------------------ First section: import statements -----------------
import tkinter as tk

# ------------------ Second section: class and methods ----------------
class Reversi:
    def __init__(self):
```

Creates the main window, playing grid, four starting discs and interactive buttons (Quit, Reset, Instructions, Choose Colors!).

def chooseColor(self):

Creates a new window that allows players to choose their disc and grid colors.

def updateColor(self):

Changes the color of the players' discs and grid to what they have chosen.

def whoseTurn(self):

Determines whose turn it is to make a move.

def placeDisc(self):

Creates and places discs on playable (empty) squares on the grid; checks if the game is over (all squares filled).

def mapToGrid(self, realex, realey):

Converts disc coordinates on canvas to index positions in a nested list of disc colors.

def discFlipper(self, a, b):

Checks surrounding discs and flips discs according to game rules.

def gameOverSequence(self):

Ends gameplay and creates a game over window with players’ scores and options to replay or quit.

def replay(self):

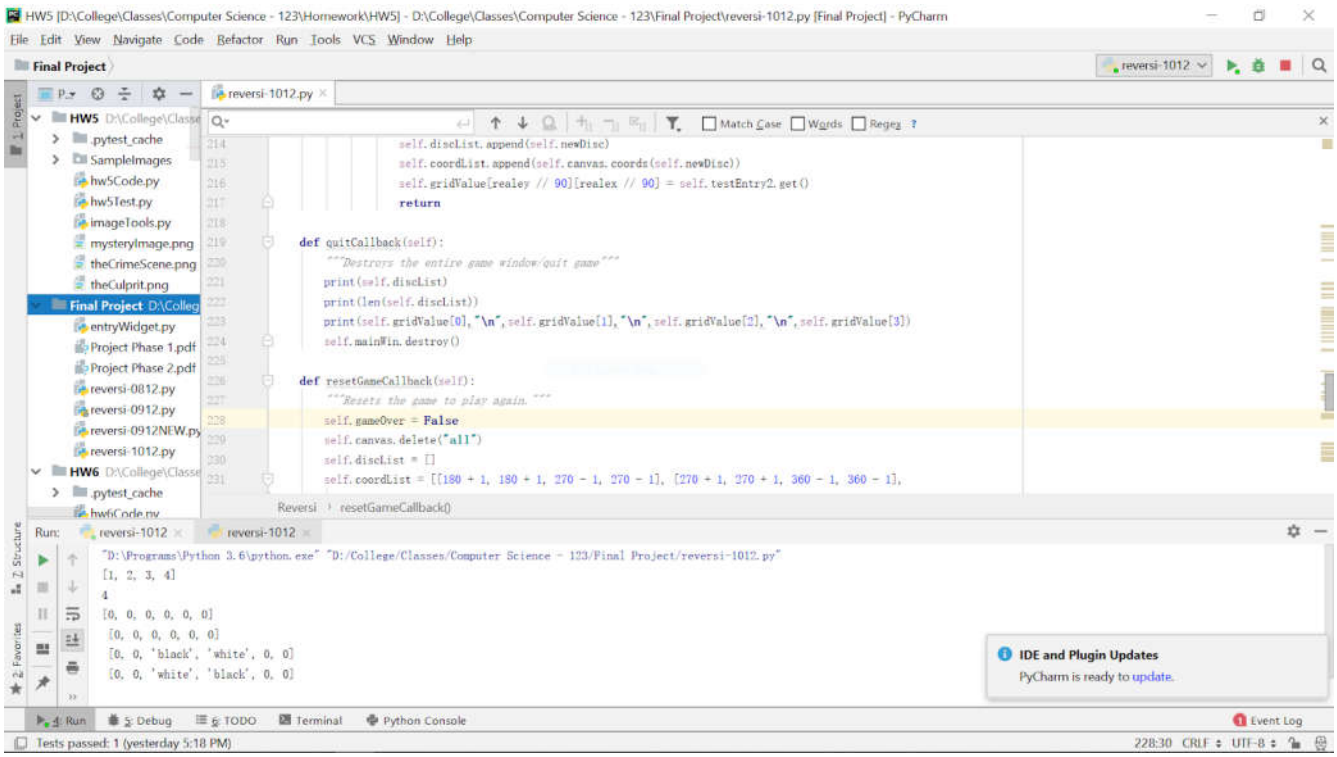
```
        Resets game variables to a new-game state, and calls
        resetGameCallback.

        def resetGameCallback(self):
        Resets the game to play again.

        def instructionsCallback(self):
        Creates a window that shows instructions for how to use our program
        and play the game.

        def quitCallback(self):
        Destroys the main window and quits the game.

        def run(self):
        Runs the GUI.

# ------------------ Third section: main program ----------------------

myReversi = Reversi()
myReversi.run()
```

❖ **Implementation and Testing**

As we did our project using `tkinter` and GUI, we were unable to write test files for it. Instead, we added many print functions to test whether each line worked, as we could see the printed statement in the console.

In the example screenshot of our code below, we have made the program print out a nested list that reflects the value (color) of every disc currently on the grid when the 'Quit' button is pressed, setting empty squares to '0' and squares with discs to the disc's color. We used print statements such as this one to easily test the logic of our program, especially with complicated methods such as **discFlipper**. Similar

test calls and print statements have been left for inspection in our submitted code file, but commented out.



We also played the game many times to check if the discs were changing color as they should be, both when we chose our own colors and when the discs were set to their default color schemes. We read through our code line by line in the buggy methods, compared what we expected them to do with the printed statements showing what they actually did, and edited the buggy lines, replaying the game to see their effects.

The **discFlipper** method should be the only part of our program that may contain bugs. Unfortunately, we cannot check every possible move, so there might be some problems we did not find out. We are confident with the other parts of our program.

❖ Reflection

Our project is based on `tkinter` and GUIs. During the process, we learned a lot more about GUIs and object-oriented programming as we needed to use more methods and attributes than what we were taught in class. Our original plan was to make our game as close to Reversi as possible but we could not figure

out how to change the disc colors when more than one disc is trapped, so we will try to examine the logic of our programs more carefully in future coding exercises.

It is surprising that though we did not actually flip the discs, we found a way to create a new disc on top of the original one and this has the same effect as flipping. We were also surprised that our simple method (using if, True and False statements) for determining whose turn it was to play worked well, as we considered many approaches to alternating player turns that were far more complicated. Other parts of our program, such as creating the canvas for the playing grid and the discs as canvas shapes, worked exactly as we expected them to and required little debugging.

As an extension of our game, we were thinking of adding background music. We did not have time to do so, however, and we have not learned how to use the module that would allow this so we were unable to integrate it with our current project. We would also like to add an AI mode, but that would require more time and planning on our part to figure out the logic.

Bibliography and Resources

Fox, S., *The tkinter module for Graphical User Interfaces (GUIs)*. Retrieved December 14, 2018, from https://docs.google.com/document/d/1S6xWzzMEF30FI0WBbxxZTI4m335kiA_wEVvuLztkNfE/edit

Jokosare, S.L., (n.d.). Reversi - Game Rules. Retrieved December 14, 2018, from http://www.ludoteka.com/reversi-en.html

Lundh, F. (n.d.). Toplevel Window Methods. Retrieved December 14, 2018, from http://effbot.org/tkinterbook/wm.htm

Lundh, F. (n.d.). Widget Styling. Retrieved December 14, 2018 from http://effbot.org/tkinterbook/tkinter-widget-styling.htm

Python Color Constants Module. (2015, March 20). Retrieved December 14, 2018, from https://www.webucator.com/blog/2015/03/python-color-constants-module/?fbclid=IwAR189ctDhSVqUlu6ft2wCsSdXiQ0VcK8wjjUPiY26RobYrz3Gdz_8N5TMjk